ARCOS-Speculus Detailed specification

1. Purpose and Role

The purpose of the ARCOS-Speculus is to define the Domain to use and their specific AI Agents to use. It provides the Orchestrator with the necessary information as to know how to call each Domain related Component (Speculus, Producer, Validator, Post-Processor, InputFilter)

It is one of the two parts of the ARCOS architecture that interacts with the client, the other is the Domain Speculus.

- The Orchestrator, the Producer, Validator, PostProcessor, InputFilter are not interacting with the user.
- Some clarification requests are sent to the orchestrator that are then being sent to the Domain Speculus to extract some rules from user clarification.

Upon modification of any selected AI Agent, a complete reset of the project occurs. If only Purpose or some other elements are changed, the process can continue as was and memory kept.

Two choices to get out of the ARCOS Speculus … Cancel or Save and Proceed. The choice of the User is sent back to the Orchestrator that then decides what's the next step. The Domain Speculus also sends the ARCOS_Project.xml file containing all the specified AI agent to use in the project.

2. Process Flow

First select a domain for which there are available AI Agents. A domain can have many versions in their offering of each component needed for the domain.

Once the domain is selected, it is up to the user to select, for each component of the ARCOS architecture, the versions of the available AI agents (Domain Speculus, Producer, Validator, Post-Processor, FilterInput).

> Right now there are no list of available domains and their AI Agents offering so I need a Path or URL, a User, a Password, and Args to get to the parts I'll be using for the demo.

Also defined in the ARCOS-Speculus is the domain schema … for the BLEU example, the domain schema is the part schema for a manufacturer to upload their inventory to be sold by BLEU. The Domain is BLEU with only one offering, building a Rust CRUD interface library to be included in their manufacturing software to build the BLEU sendoff.

You can also include some Constraints for performance and security.

Once everything is defined, the user can either cancel, or save and start the Domain loop, sending the ARCOS_Project.xml to the Orchestrator to start the loop.

ARCOS-Speculus serves to define the project specification before orchestration begins.

3. Communication between ARCOS-Speculus and the Orchestrator (Maestro) using schema-based XML.

How to get an agnostic orchestrator and a domain specific agents work together is to have a separation of concern and verifiable communication channels. By using XML Schema for all communications between the Orchestrator and its component, we assure that they are verifiable against a defined standard that allows the domain information to be passed around to the domain related AI Agents and keep the orchestrator agnostic.

Schemas

Maestro_ARCOS_Speculus_Messaging.xsd

<ARCOSSpeculusMessage>		
NewProject	Choice	
EditProject	Choice	
Checksum	Common:SHA256Type	

ARCOS_Speculus+Response.xsd

<ARCOSSpeculusResponse>		
<Success> OR	Choice	
<ProjectID>		
<SavedAt>		
<ARCOS_Project>	inline	
<Summary>		
<Checksum>		
<Timestamp<		
<Cancelled>	Choice	
<ProjectID>		
<Reason>		Optional
<Timestamp>		

The ARCOS_Project.xml contains the Domain Specific information needed by the Orchestrator to load the correct domain specific components needed by the domain. The Orchestrator remains Agnostic.

4. Elements and Attributes

This is the element <ARCOSProject> and its main components.

Element / Attribute	***Description***	***Usual Values / Notes***
<ARCOSProject>	Root of project specification	Required
<ProjectID>	Unique project identifier	string
<Purpose>	Project purpose	Free text string
<DomainReference>	Defines domain schema	Attributes: xsd (required, schema location), version (optional).
<DomainAgents>	Lists the agents for this project.	
<DomainSpeculus>	Defines the domain-specific speculator agent.	
<Producer>	Defines the Producer agent.	
<Validator>	Defines the Validator agent.	
<PostProcessor>	Post-Processor agent.	Optional
<FilterInput>	Pre-processor/filter agent	Optional
<Licensing>	Licensing information	Contains mode, validation, license duration
<Constraints>	Project constraints	Performance, Security, flags for realtime/embedded/portable

The <DomainReference> has the following attributes

@xsd	Schema location to use for the domain	Required
@version	Version of it	Optional

Each Domain Agents have the following elements and attributes

<%%%Agent%%%>	DomainSpeculus, Producer, Validator, PostProcessor, FilterInput
@role	DomainSpeculus, Producer, Validator, PostProcessor, FilterInput
<Execution>	
@type	app or api
<Path> or	Local executable path.
<URL>	Remote API endpoint
<User>	Optional credential
<Password>	Optional credential
<Args>	Optional command line arguments

The <Licensing> tag has the following elements and attributes

@mode		
<Mode>		
<Validation>		
<LicenseDuration>		

The <Constraints> tag has the following elements and attributes

@realtime	Boolean	
@embedded	Boolean	
@portable	Boolean	
<Performance>		
@level		
<Security>		
@encryption		
@policy		